

FIG. 1

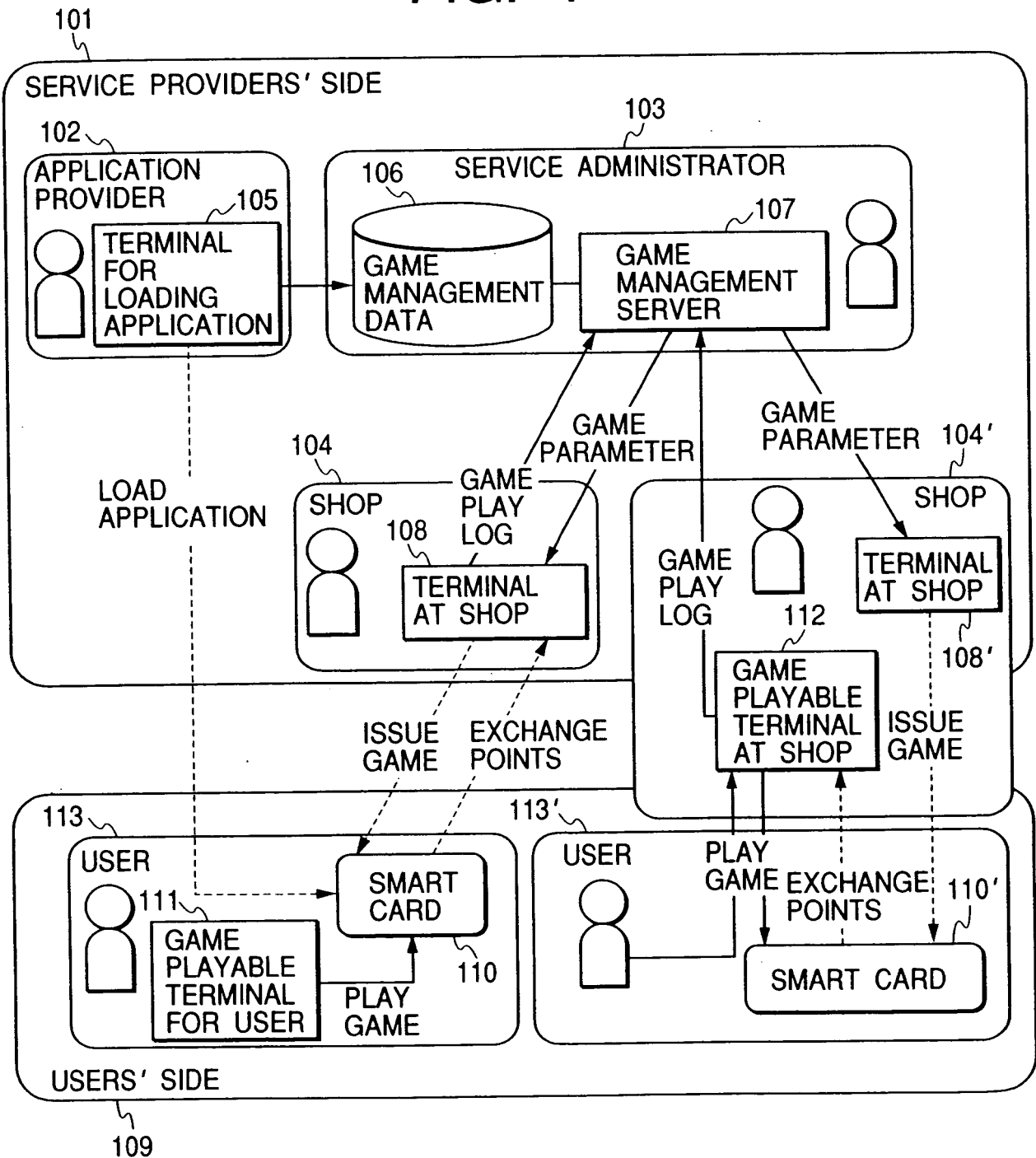


FIG. 2

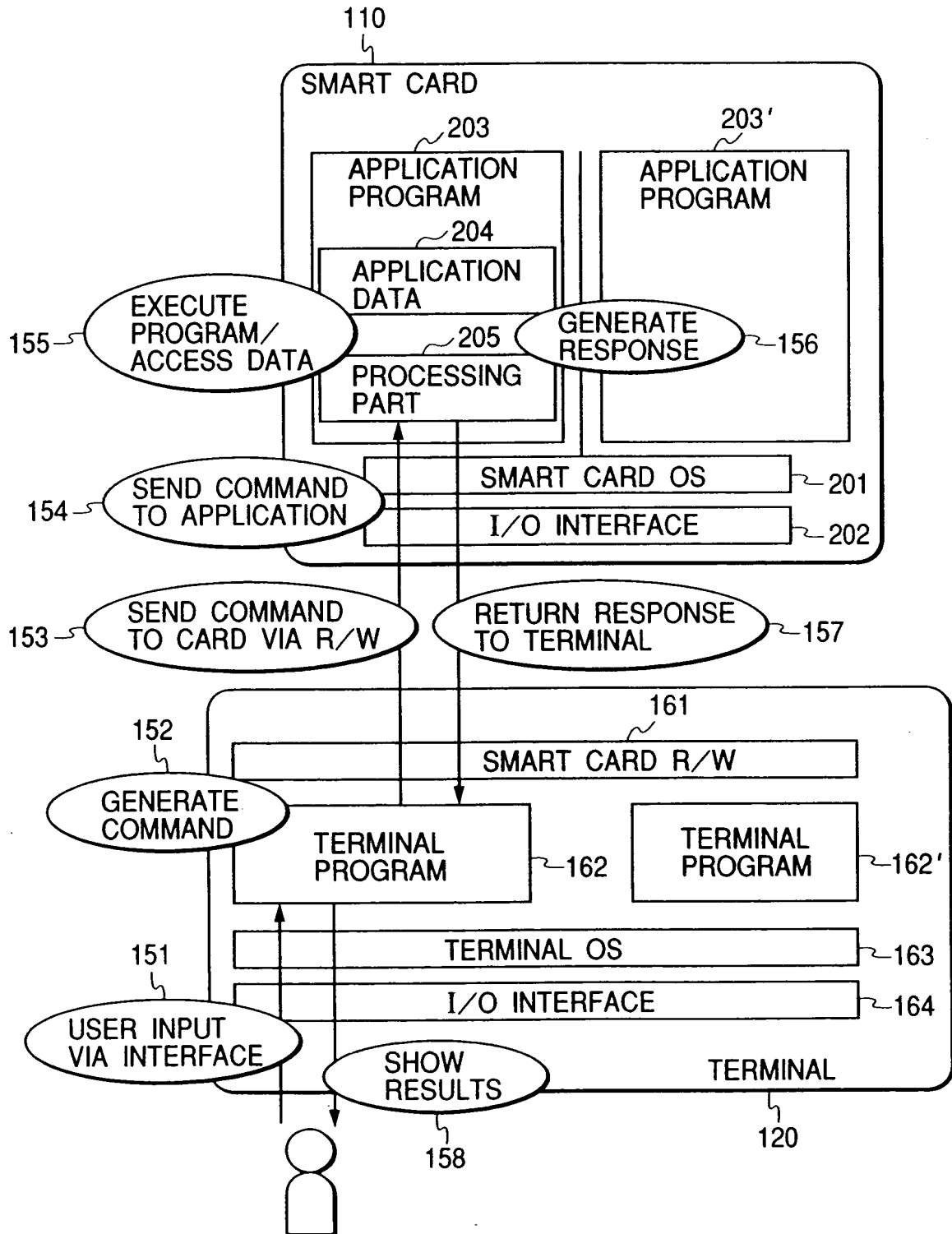


FIG. 3

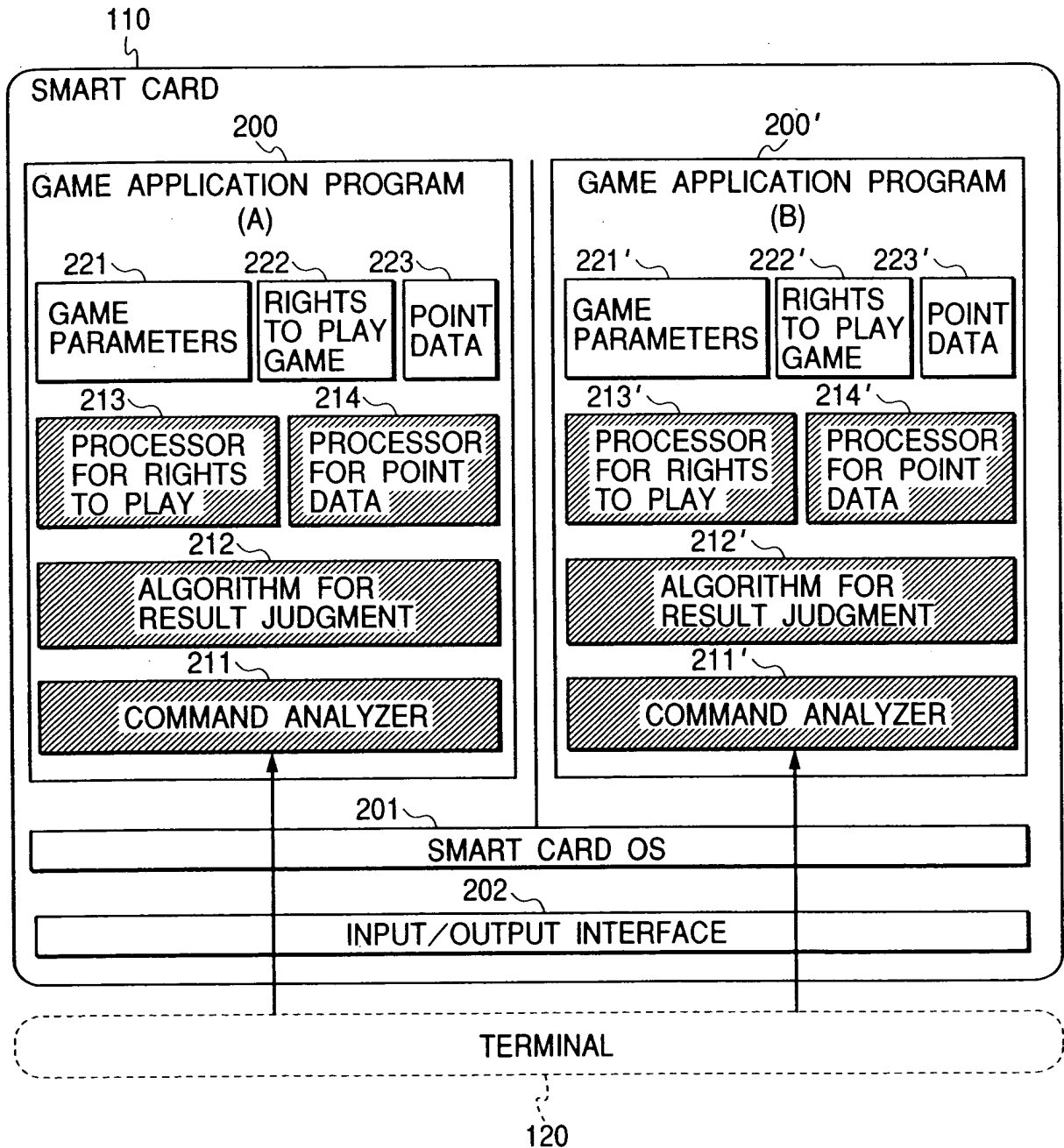


FIG. 4

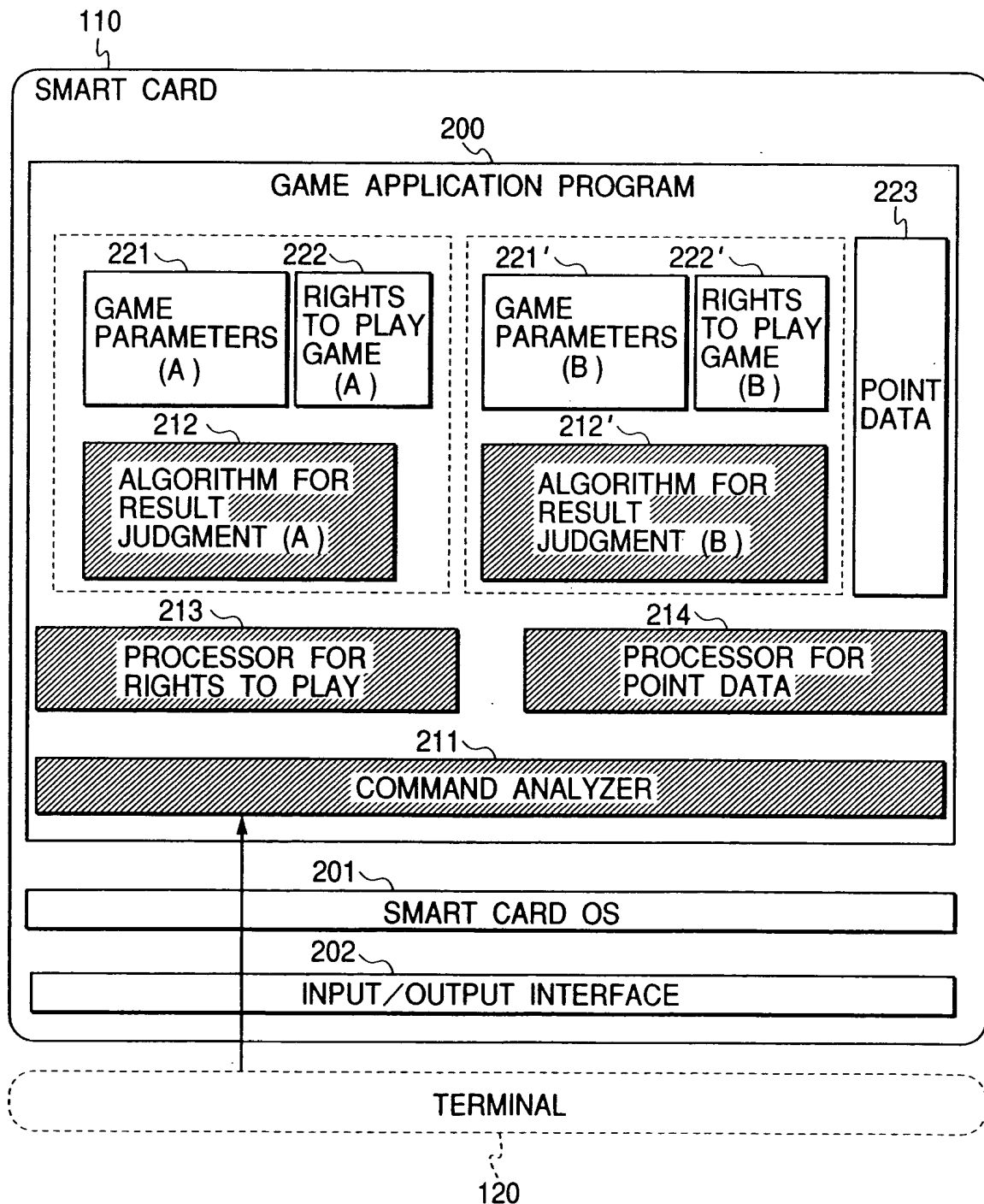
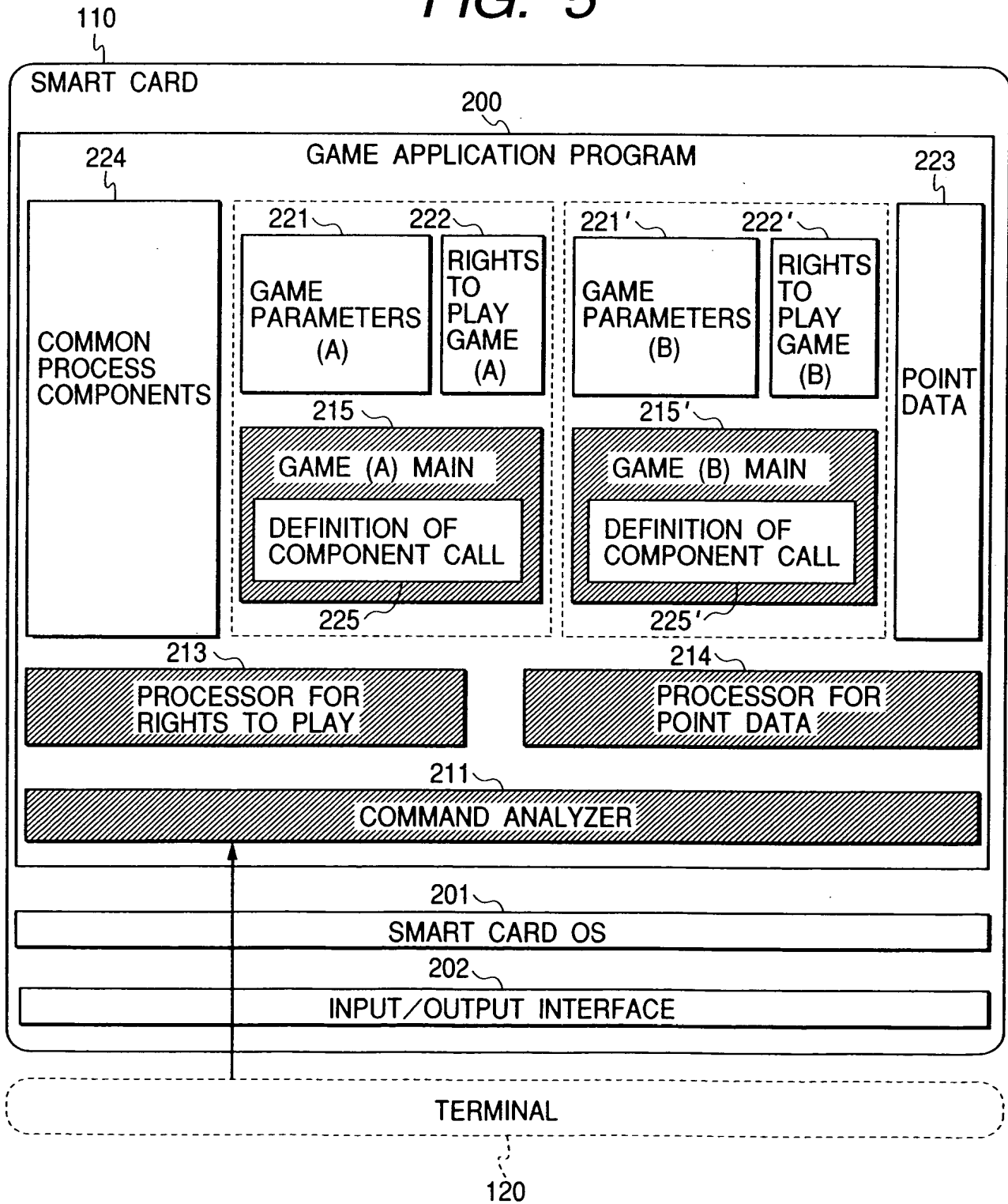
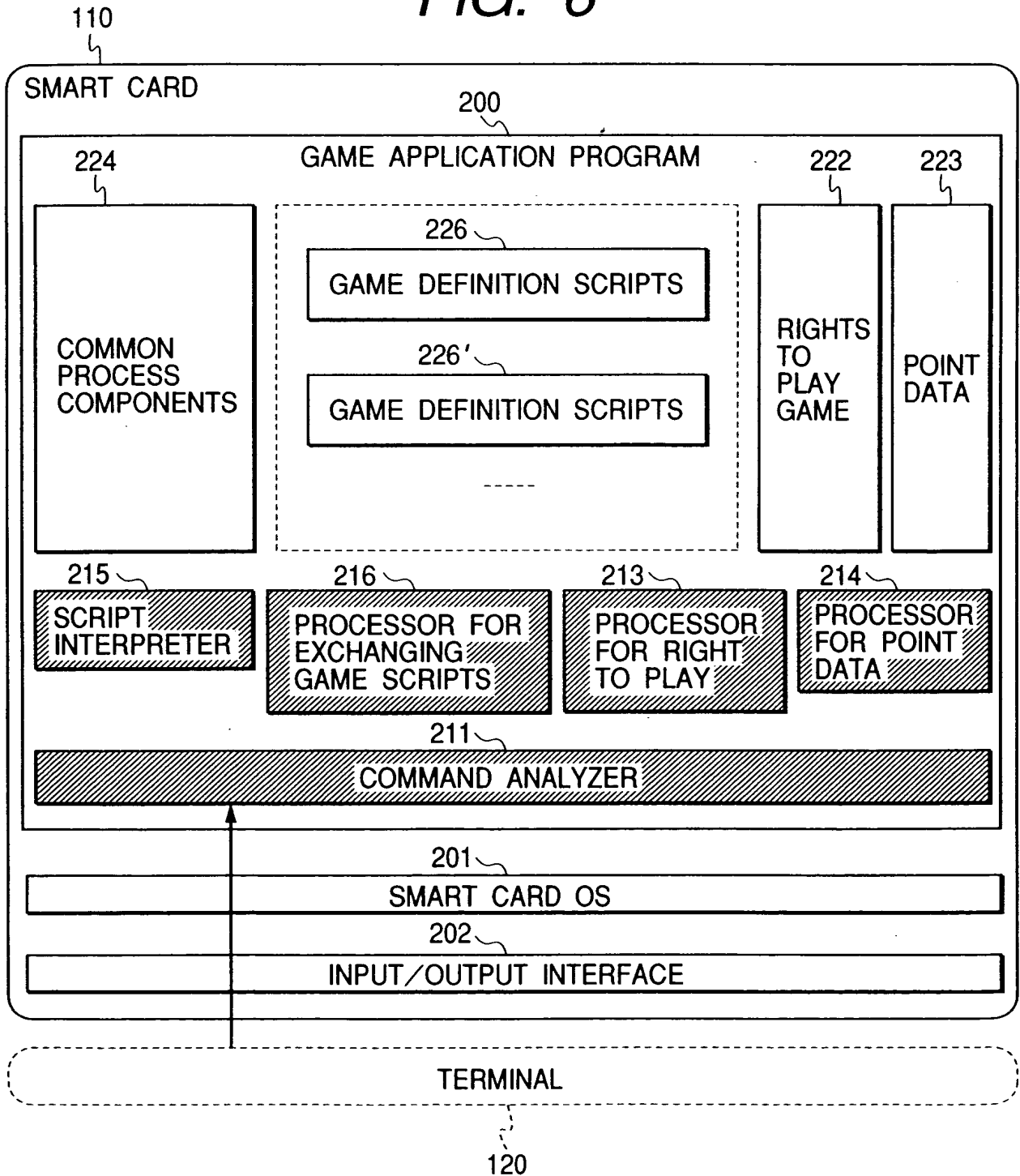
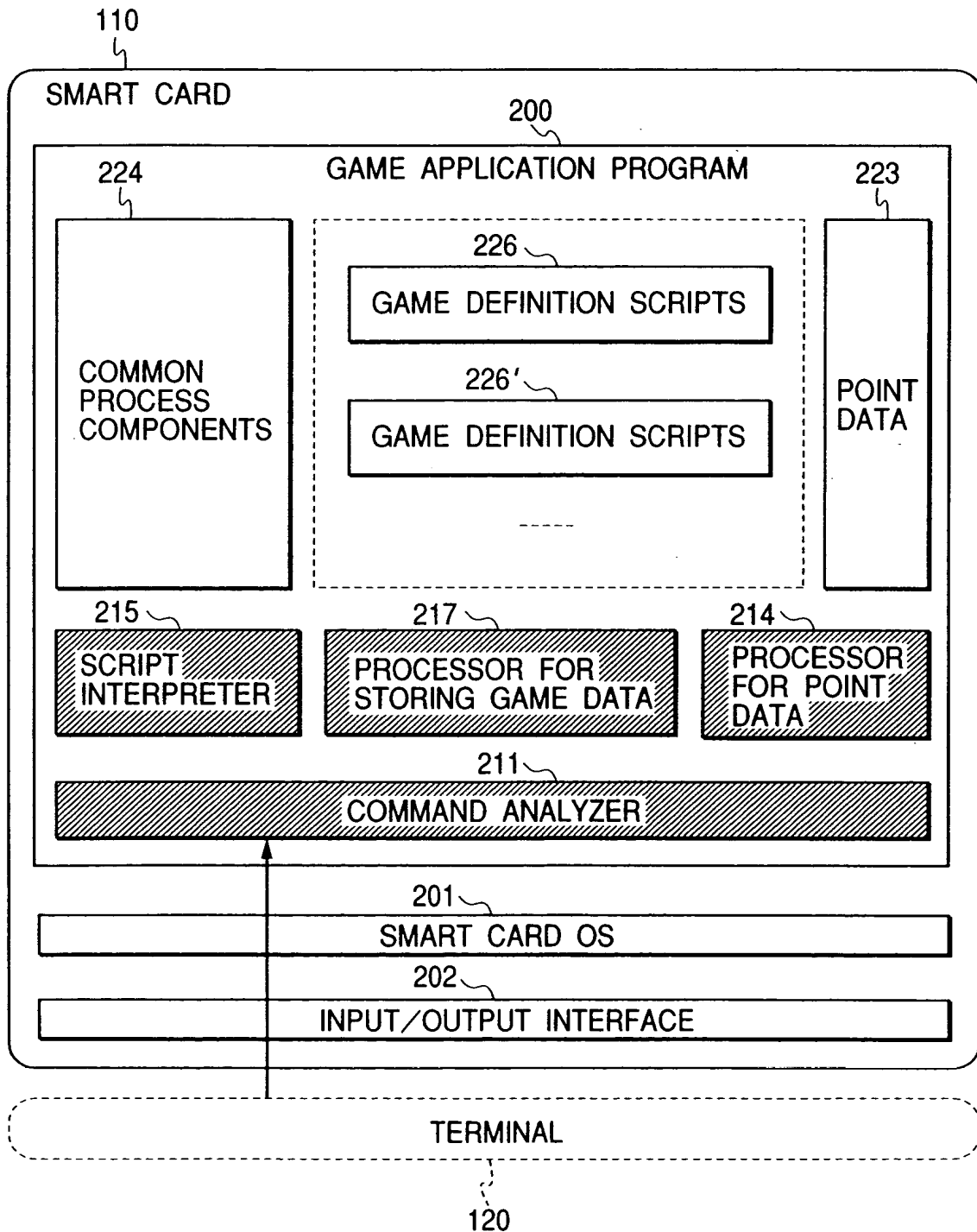


FIG. 5



**FIG. 6**

**FIG. 7**

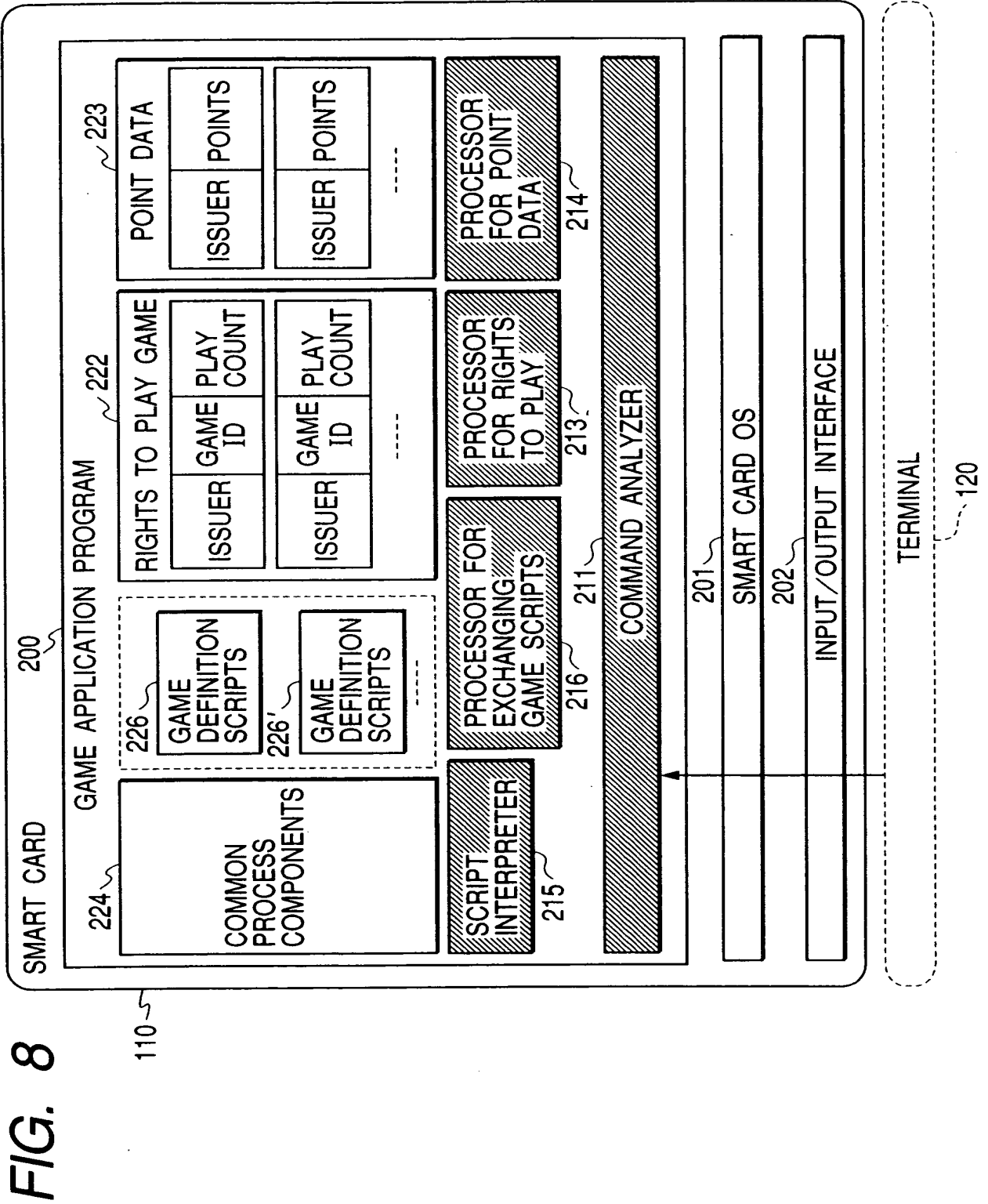
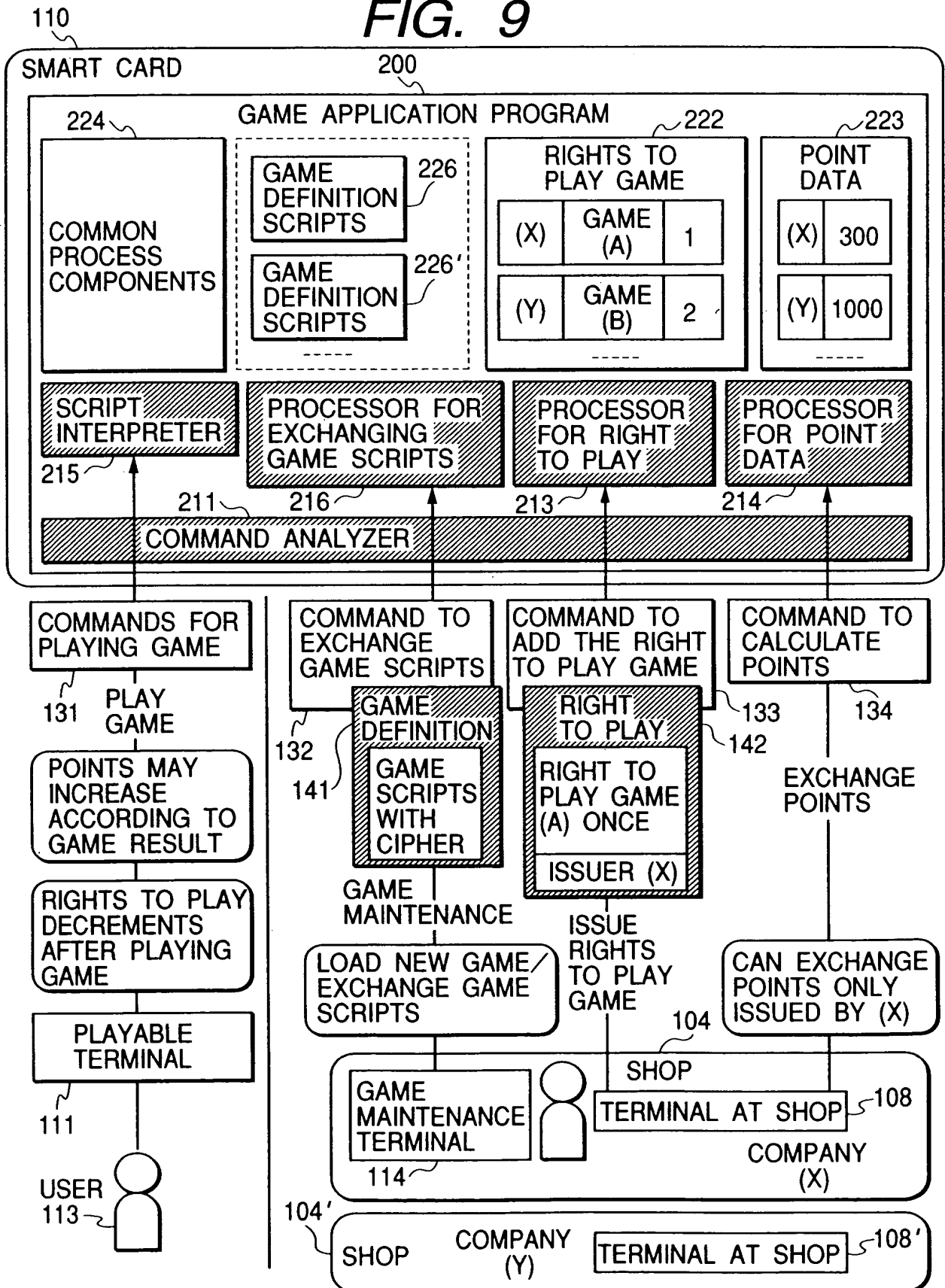


FIG. 9



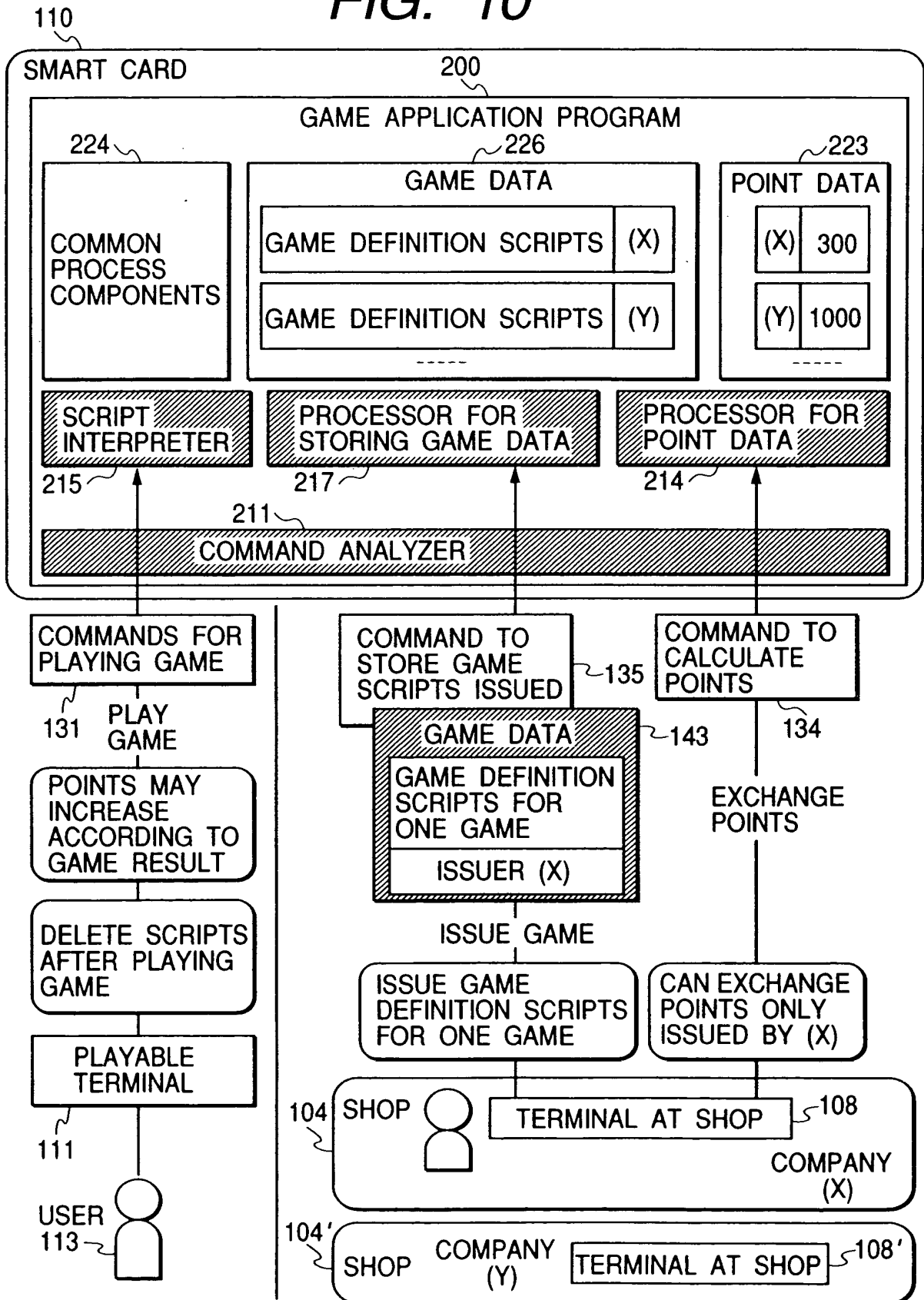
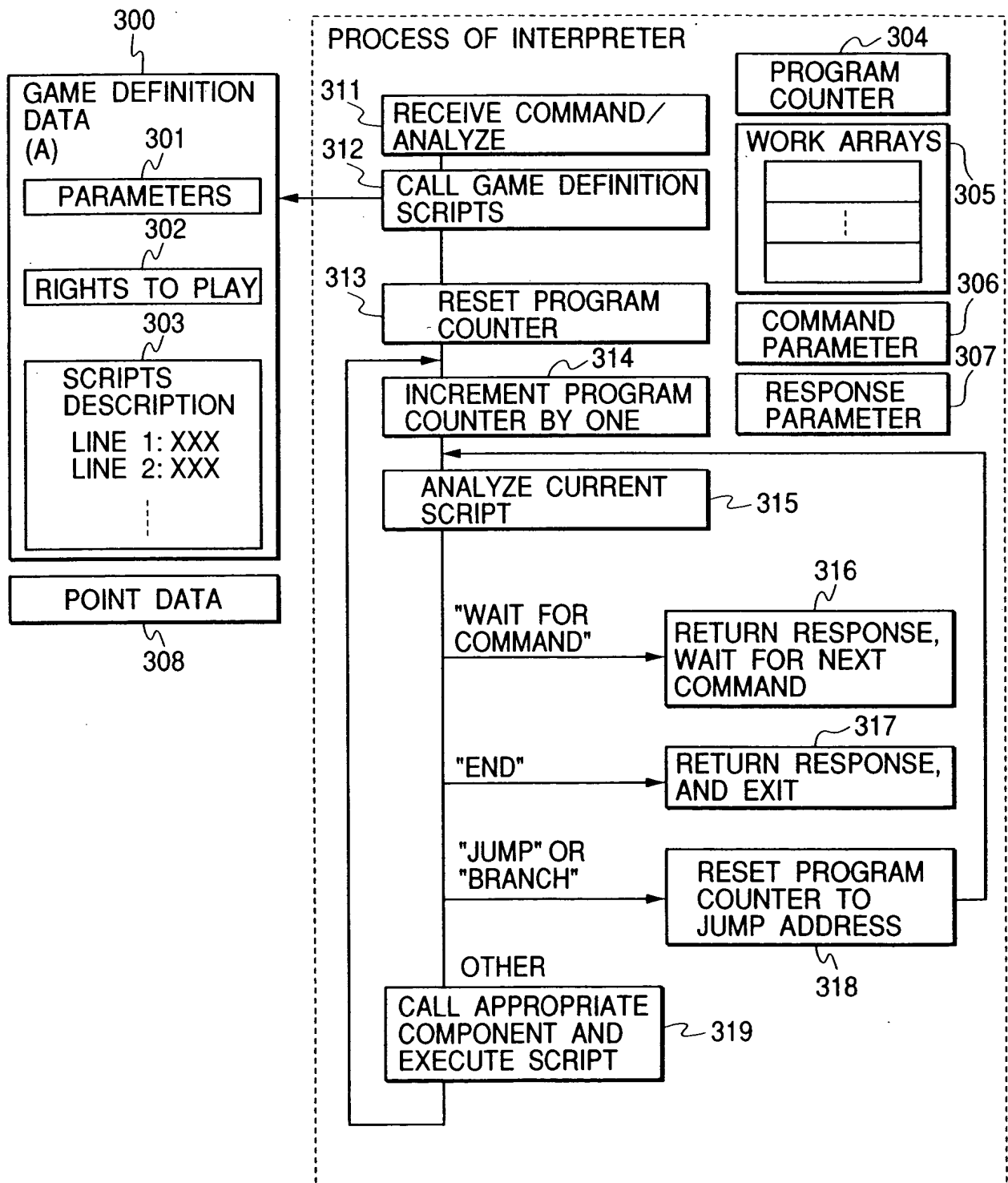
**FIG. 10**

FIG. 11



*FIG. 12*

※ array[]: work array (305)

cmd[] : command parameter (306)

rsp[] : response parameter (307)

[store data]

set (a, b)      Store value b into array[a]

random (a, b)      Generate a random number (0 to b) and store it into array[a]

getcmd (a, b)      Store value of cmd[b] into array[a]

setrspa (a, b)      Store the contents of array[a] into rsp[b]

setrspv (a, b)      Store value a into rsp[b]

getcmda (a, b)      Store the contents of array[(cmd[b])] into array[a]

[addition/subtraction and compare]

adda (a, b, c)      Add the contents of array[b] and the contents of array[c] and store the result into array[a]

suba (a, b, c)      Subtract the contents of array[c] from the contents of array[b] and store the result into array[a]

cmpa (a, b, c)      Compare array[b] and array[c] contents and store the result into array[a]

(0×00: equal, 0×01: yy>zz, 0×02: yy<zz)

addv (a, b, c)      Add the contents of array[b] and the value of c and store the result into array[a]

subv (a, b, c)      Subtract the value of c from the contents of array[b] and store the result into array[a]

cmpv (a, b, c)      Compare the contents of array[b] and the value of c and store the result into array[a]

(0×00: equal, 0×01: yy>zz, 0×02: yy<zz)

[jump, branch]

jmp (a)      Jump to line a

eq (a, b, c)      Jump to line c if the contents of array[a]= the value of b

ne (a, b, c)      Jump to line c if the contents of array[a]≠ the value of b

[point calculation]

pointa (a)      Add the contents of array[a] to point data

pointv (a)      Add value a to point data

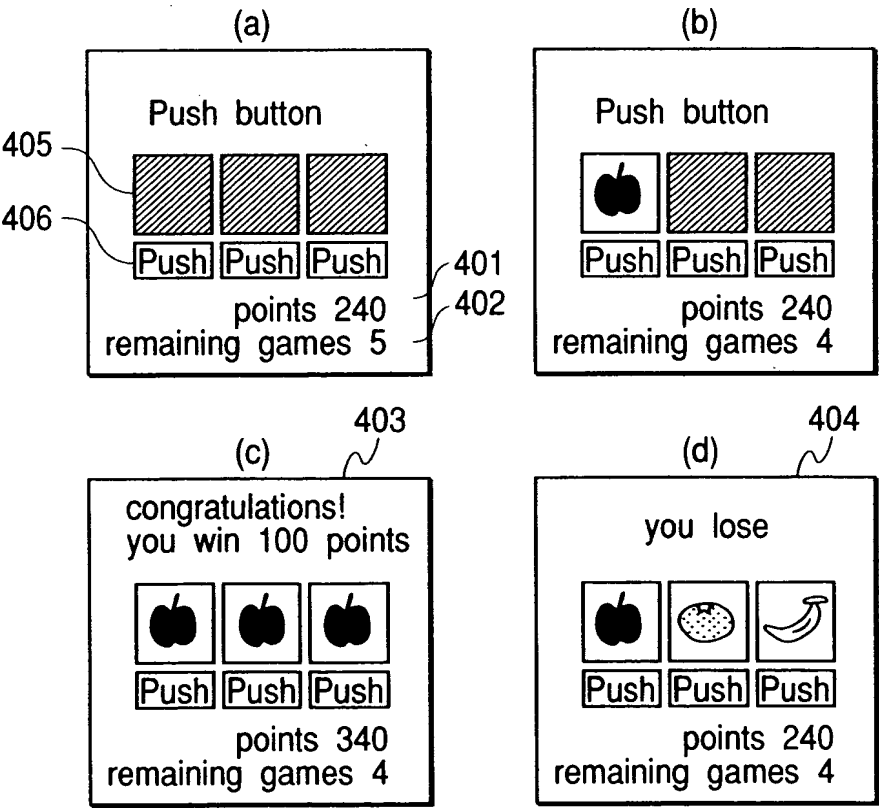
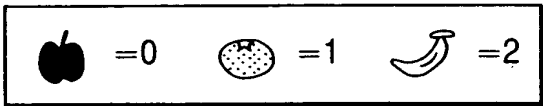
[return response to terminal]

return      Return rsp[] as response data, and wait for next command

end      Return rsp[] as response data, and exit from game script execution

**FIG. 13**

SLOT MACHINE: BOX NUMBER=3, RANDOM NUMBER=0 TO 2

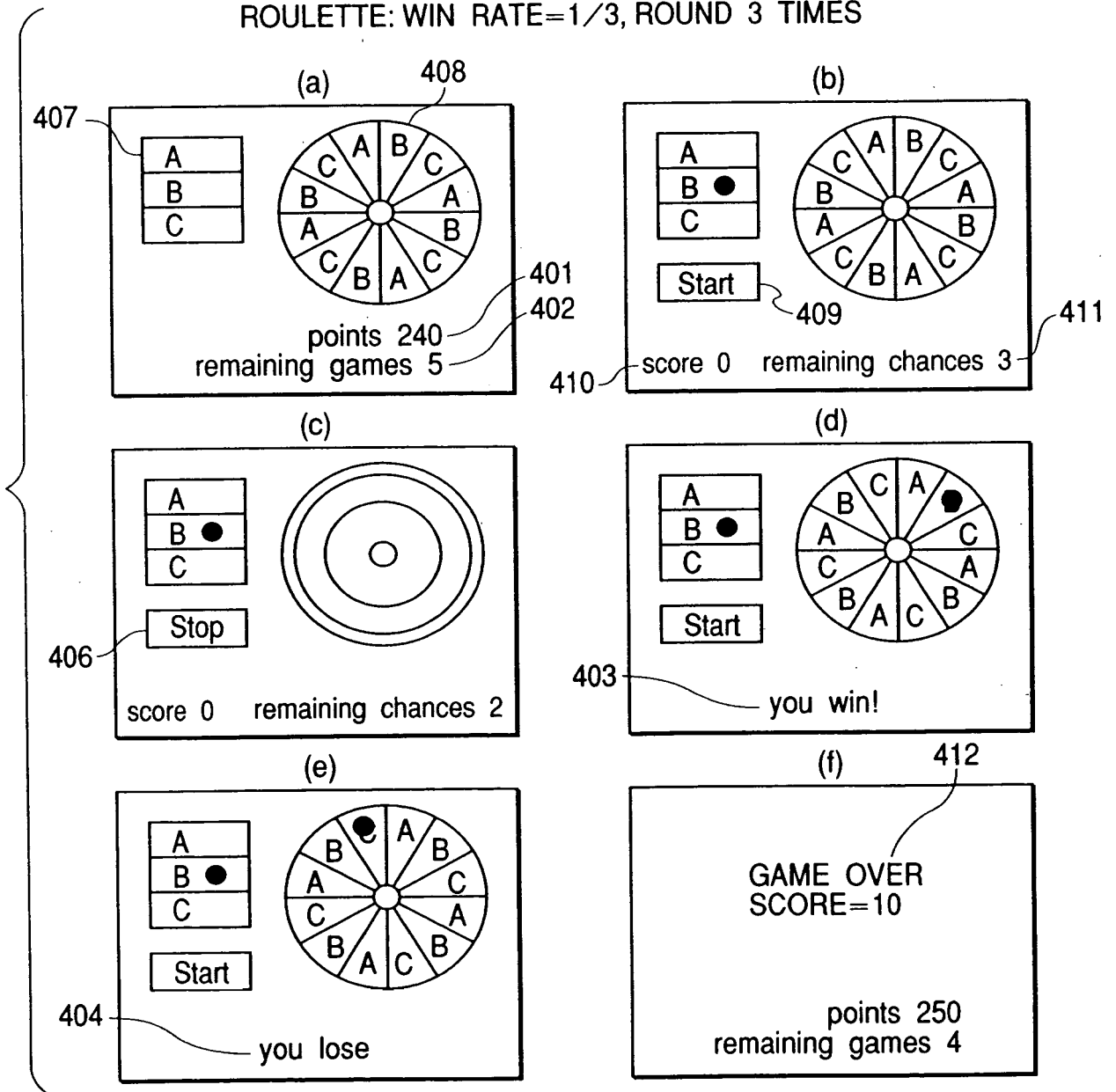


*FIG. 14*

LINE	SCRIPTS	COMMENT
00	random (0, 2)	Store a random number (0 to 2) into array[0]
01	setrspa (0, 0)	Store the contents of array[0] into rsp[0]
02	return	Return response, and wait for command
03	random (1, 2)	Store a random number (0 to 2) into array[1]
04	setrspa (1, 0)	Store the contents of array[1] to rsp[0]
05	return	Return response, and wait for command
06	random (2, 2)	Store a random number (0 to 2) into array[2]
07	setrspa (2, 0)	Store the contents of array[2] into rsp[0]
08	setrspv (1, 1)	Store a value of 1 (to indicate the lost) into rsp[1]
09	cmpa (3, 0, 1)	Compare array[0] and array[1] contents and store the result into array[3]
0a	cmpa (4, 0, 2)	Compare array[0] and array[2] contents and store the result into array[4]
0b	eq (3, 0, 0d)	Jump to line 0×0d if array[3] contains 0
0c	end	Return response, and exit
0d	eq (4, 0, 0f)	Jump to line 0×0f if array[4] contains 0
0e	end	Return response, and exit
0f	pointa (50)	Add 50 to point data
10	setrspv (2, 50)	Store 50 (points won at this game) into rsp[2]
11	end	Return response, and exit

**FIG. 15**

ROULETTE: WIN RATE=1/3, ROUND 3 TIMES



*FIG. 16*

LINE	SCRIPT	COMMENT
00	getcmd (0, 0)	Store the value of cmd[0] into array[0]
01	random (1, 2)	Store a random number (0 to 2) into array[1]
02	cmpa (2, 0, 1)	Compare array[0] and array[1] contents and store the result into array[2]
03	setrspa (1, 0)	Store the contents of array[1] into rsp[0]
04	setrspa (2, 1)	Store the contents of array[2] into rsp[1]
05	return	Return response, and wait for command
06	getcmd (0, 0)	Store the value of cmd[0] into array[0]
07	random (1, 2)	Store a random number (0 to 2) into array[1]
08	cmpa (3, 0, 1)	Compare array[0] and array[1] contents and store the result into array[3]
09	setrspa (1, 0)	Store the contents of array[1] into rsp[0]
0a	setrspa (2, 1)	Store the contents of array[2] into rsp[1]
0b	return	Return response, and wait for command
0c	getcmd (0, 0)	Store the value of cmd[0] into array[0]
0d	random (1, 2)	Store a random number (0 to 2) into array[1]
0e	cmpa (4, 0, 1)	Compare array[0] and array[1] contents and store the result into array[4]
0f	setrspa (1, 0)	Store the contents of array[1] into rsp[0]
10	setrspa (2, 1)	Store the contents of array[2] into rsp[1]
11	setrspv (1, 2)	Store a value of 1 (to indicate the last) into rsp[1]
12	ne (2, 0 14)	Jump to line 0×14 if the contents of array[2]≠0
13	set (5, 20)	Store 20 (points won per game) into array[5]
14	ne (3, 0, 16)	Jump to line 0×16 if the contents of array[3]≠0
15	addv (5, 5, 20)	Add 20 (points won per game) to array[5]
16	ne (4, 0, 16)	Jump to line 0×18 if the contents of array[4]≠0
17	addv (5, 5, 20)	Add 20 (points won per game) to array[5]
18	pointv (5)	Add the contents of array[5] to points
19	setrspa (5, 2)	Store the contents of array[5] (final points) into rsp[2]
1a	end	Return response, and exit

**FIG. 17**

SHOOTING GAME: TARGET NUMBER=5, ROUND NUMBER=5

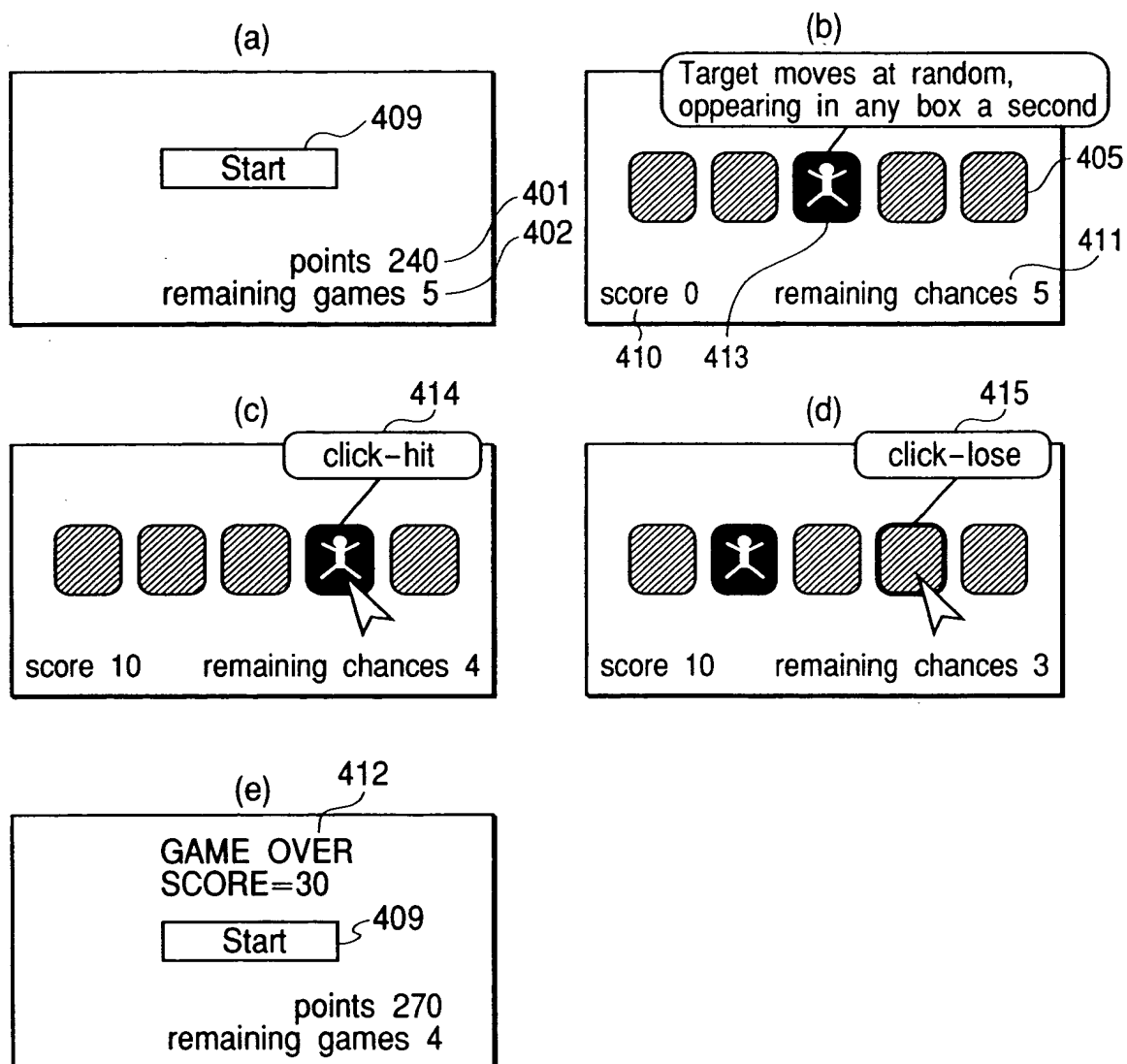
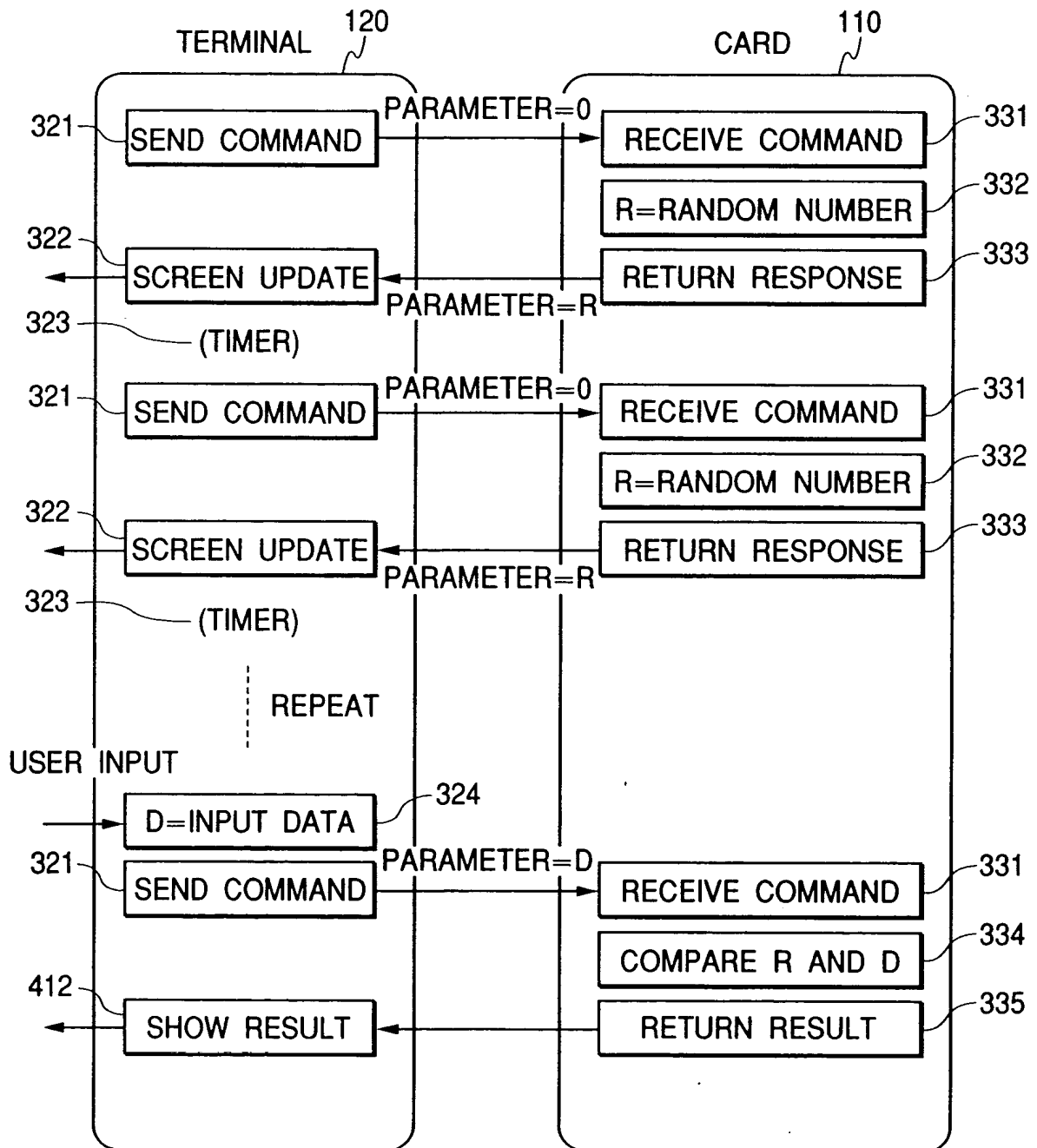


FIG. 18



**FIG. 19**

LINE	SCRIPT	COMMENT
00	set (0, 0)	Store 0 into array[0] (loop index)
01	set (1, 0)	Store 0 into array[1] (points won)
02	set (2, 0)	Store 0 into array[2] (random number to be compared with input)
03	getcmd (3, 0)	Store the value of cmd[0] into array[3]
04	eq (3, 0, 12)	Jump to line 0×12 if array[3] contents 0 (Comparing values follows)
05	cmpa (4, 2, 3)	Compare array[2] and array[3] contents and store the result into array[4]
06	setrspa (4, 0)	Store the contents of array[4] (result of comparison) into rsp[0]
07	addv (0, 0, 1)	Increment the counter value contained in array[0] by one
08	setrspa (0, 1)	Store the contents of array[0] (loop count) into rsp[1]
09	ne (4, 0, 0b)	Jump to line 0×0b if the contents of array[4]≠0 (means loosing)
0a	addv (1, 1, 10)	Add 10 (additional points) to array[1]
0b	eq (0, 5, 0e)	Jump to line 0×0e if the contents of array[0]=5 (last)
0c	return	Return response and wait for command
0d	jump (03)	Jump to line 0×03 (beginning of loop) (Last loop execution follows)
0e	pointv (1)	Add the contents of array[1] to points
0f	setrspa (1, 2)	Store the contents of array[1] (points) into rsp[2]
10	setrspv (1, 3)	Store a value of 1 (to indicate the last) into rsp[3]
11	end	Return response and exit (Generating a random number follows)
12	random (2, 5)	Store a random number (0 to 4) into array[2]
13	addv (2, 2, 1)	Increment the value contained in array[2] by one (the range of random numbers change to 1 to 5)
14	setrspa (2, 0)	Store the contents of array[1] (random number) into rsp[0]
15	return	Return response and wait for command
16	jump (0, 3)	Jump to line 0×03 (beginning of loop)

FIG. 20

